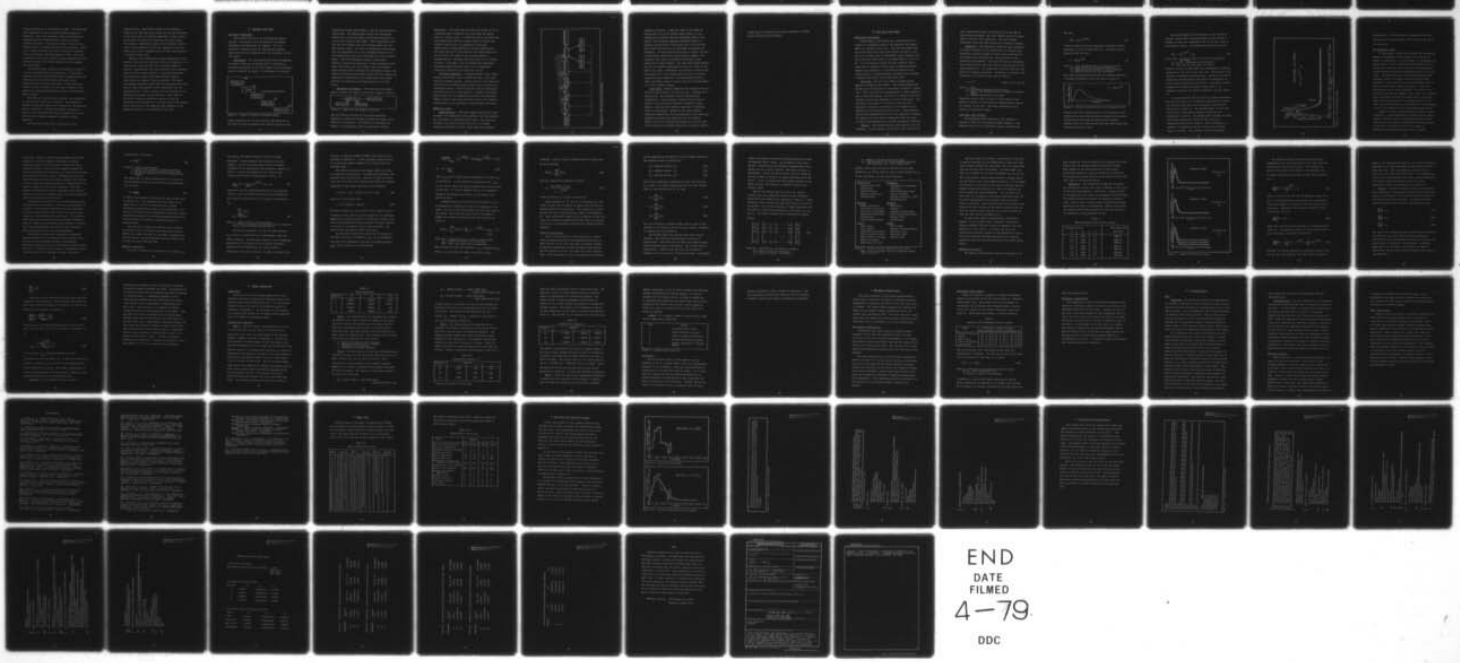
AD-A064 223 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/6 5/3
AN APPROACH TO SOFTWARE LIFE CYCLE COST MODELING.(U)
DEC 78 W H WALKER

UNCLASSIFIED AFIT/GCS/EE/78-21 NL

| OF |
AD
A064223

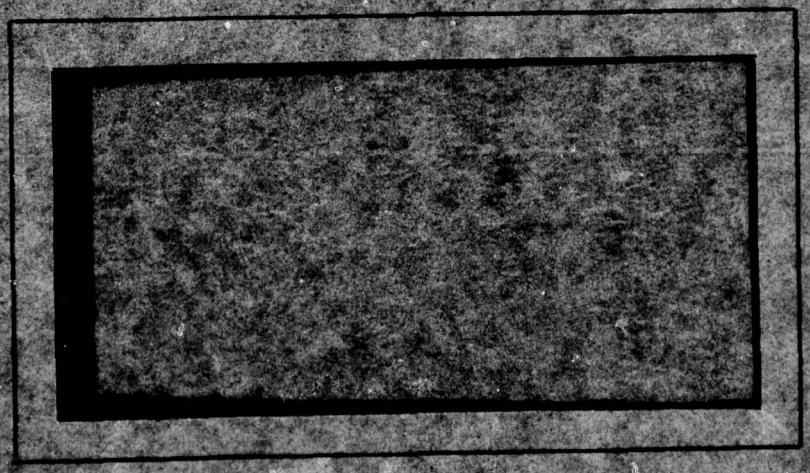


END
DATE
FILMED
4-79
DDC

AIR FORCE INSTITUTE OF TECHNOLOGY

LEVEL IV

AIR UNIVERSITY
UNITED STATES AIR FORCE



SCHOOL OF ENGINEERING

WRIGHT-PATTERSON AIR FORCE BASE, OHIO

79 01 30 116

AFIT/GCS/EE/78-21

(1)

LEVEL II

An Approach to

SOFTWARE LIFE CYCLE COST MODELING

THESIS

AFIT/GCS/EE/78-21 William H. Walker IV
Capt USAF

Approved for Public Release; distribution unlimited.

(14)

AFIT/GCS/EE/78-21

(6)

An Approach to SOFTWARE LIFE CYCLE COST MODELING.

(9) Master's theses,

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air Training Command
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by

(10) William H. Walker, IV, ~~B.S.~~
Capt USAF

Graduate Computer Systems

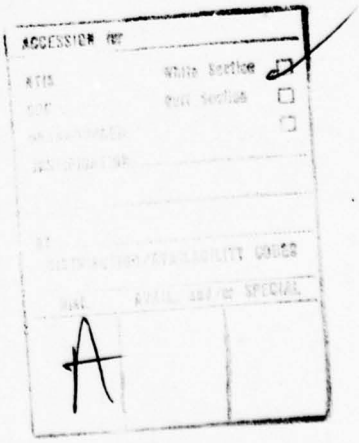
(11) December 1978

(12) 76p

Approved for Public Release; distribution unlimited.

Preface

Although I claim the credit or discredit that results from the research value of this thesis, there are many others who deserve credit for prodding and priming the effort. In particular, the members of my thesis committee, Dr T. Regulinski, Dr T. Hartrum, and Dr C. McNichols, provided the incentive and inspiration to hurdle some of the major stumbling blocks during the development of this modeling process. My deepest appreciation goes to my wife and daughter who gave me support, time, and help to get through this program.



Contents

List of Figures

List of Tables

Abstract

This report describes the development of a software life cycle costing model. The model reduces life cycle cost to a function of three parameters which are in turn functions of a number of factors that describe the software system. A step-by-step algorithm is presented for building the model from raw data. The model is exercised as an example with a small amount of data. Sensitivity analysis is used to help select the most salient factors. Brief descriptions of management applications and recommendations are presented. Appendices describe sample data and two computer programs used to develop the model.

An Approach to SOFTWARE LIFE CYCLE COST MODELING

I Introduction

Motivation

Life cycle costing is a technique of managing systems. Decisions are made based on the long range, not just immediate, impact on cost. Operation and maintenance cost must be considered as well as cost of development and procurement. Alternatives are evaluated in terms of the resultant life cycle cost as well as technical and operational factors. The cost could be measured in dollars, time, opportunity lost, or any number of other units. Department of Defense Directive (DODD) 5000.28 defines life cycle cost as:

> "... the total cost to the government of acquisition and ownership of that system over its full life. It includes the cost of development, acquisition, operation, and, where applicable, disposal." (Ref 1:8)

There are two very important reasons for using the concept of life cycle costing. The first, and most important reason, is that life cycle costing can save money. The Department of Defense spent over three billion dollars on software in 1976 (Ref 2:41) up from one billion dollars in 1974 (ref 3:63). This figure continues to grow each year. Any technique which, for a reasonable cost, will help to control or reduce that cost should be applied when

possible.

As early as 1968, the Air Force Logistics Command used the technique of life cycle costing in procurement (Ref 1: 38-40). The contract for T-38 aircraft main landing gear tires was awarded based on the lowest bid for cost per landing. Before this life cycle cost procurement was used, the T-38s were averaging 41 landings per tire. After the award of the life cycle cost contract, the average number of landings per tire grew to 104, a 150% increase. The same technique has since been applied to electron tubes, oscilloscopes, hydraulic filters, and more with resultant cost reductions of several millions of dollars. However, the literature search for this research did not reveal a single application of life cycle cost procurement or an application of life cycle costing to procurement decisions in the case of software.

If dollar savings are not sufficient to justify the use of life cycle costing techniques, there is further pressure to develop and apply the techniques to all acquisitions. Air Force Regulation 800-11, Life Cycle Costing (LCC), directs the following:

> "The Air Force will to the maximum practical extent, determine and consider life cycle cost in the various decisions associated with the development, acquisition, and modification of defense systems and subsystems and in the procurement of components and parts." (Ref 1:16)

Knowing that life cycle costing techniques can result in significant savings and that they are required by regulation is not sufficient unless a model or methodology exists for

the application of those techniques to software acquisition.

Objectives

The objectives of this thesis effort can be divided into three highly interdependent parts. The first objective was to develop a model that would provide for derivation of the life cycle cost of a software system. To be really useful for planning and budgeting, the model should provide time phased allocations of resources over the life cycle of the system. The model was also to provide a vehicle for evaluating the effects of modern programming practices, such as structured programming and design, on the life cycle cost of software. This would be done by means of a sensitivity analysis of the resultant model.

The second and less formidable objective was to develop a computational algorithm for applying the model. The algorithm should provide a step-by-step procedure that will inexorably lead to the computation of the life cycle cost of the given software system.

The last objective was to uncover enough data to test the model and demonstrate the use of the computational algorithm. This objective was not adequately satisfied due to severe limitations on the availability of data.

Limitations

Data availability is the most severe limitation to the successful modeling of software life cycle costs. Four

factors contribute to the scarcity of data. The first and most disastrous is that the data is often simply not collected. Even the government, which is notorious for requiring massive amounts of data, does not collect software cost data. This may be a result of the cost or impracticality of separating costs into categories. After all, it does cost money and time to account for expenditures of resources, especially people's time. How would an engineer's time spent working on data communications be allocated between the software handler and the hardware bus connections?

The second factor leading to the scarcity of data is a result of competition among contractors. Proprietary interests can keep software development organizations from releasing data that they believe could give their competitors more data than they have. If the data were released, there is always the fear that it might be used against the releasing organization since the costs reveal profit margins.

When data are collected and reported, the reliability of the data must still be in question. The collection method, whether self-reported or measured over the shoulder, must be considered. Biasing must almost naturally be assumed. Raw, objective data must be sought out to avoid the effects of unknown massaging by possibly biased reporters.

The researcher must still be wary even of raw,

objective data. Consistency should be the watchword. Unless all of the data sets include the same data measured in the same units, it could be impossible to draw conclusions about the relations between the two life cycles. For instance, comparing the sizes of two software efforts would be very difficult if one was reported in lines of higher order source language while the other was reported in words of object code.

Because of the potentially extreme complexity of the software life cycle as discussed in the next chapter, it is necessary to limit the model to a tractable subset of reality. Specifically, the model addresses only technical manning of the software project in man-months per month. This limitation implies that administrative support, facilities, and computer costs are omitted. The model also ignores operating costs under the assumption that the factors under investigation affect maintenance and not operating costs. Costs to enhance or add capabilities to operational software are also omitted. These types of activities should be separately costed on their own technical merit and effect on life cycle cost of the system. Other limitations on the model are made apparent at the appropriate point in the remainder of the text.

II Software Life Cycle

Life Cycle Description

The software life cycle is an extremely complex process. The process can be divided into two phases: development and operations and support. This is a historical breakdown based on traditionally separate organizations being responsible for the two phases of the life cycle.

Development. Air Force Regulation 800-14, Management of Computer Resources in Systems (Ref 4), provides an excellent description of the five stages of the development phase as depicted in Figure 1. The concept and analysis

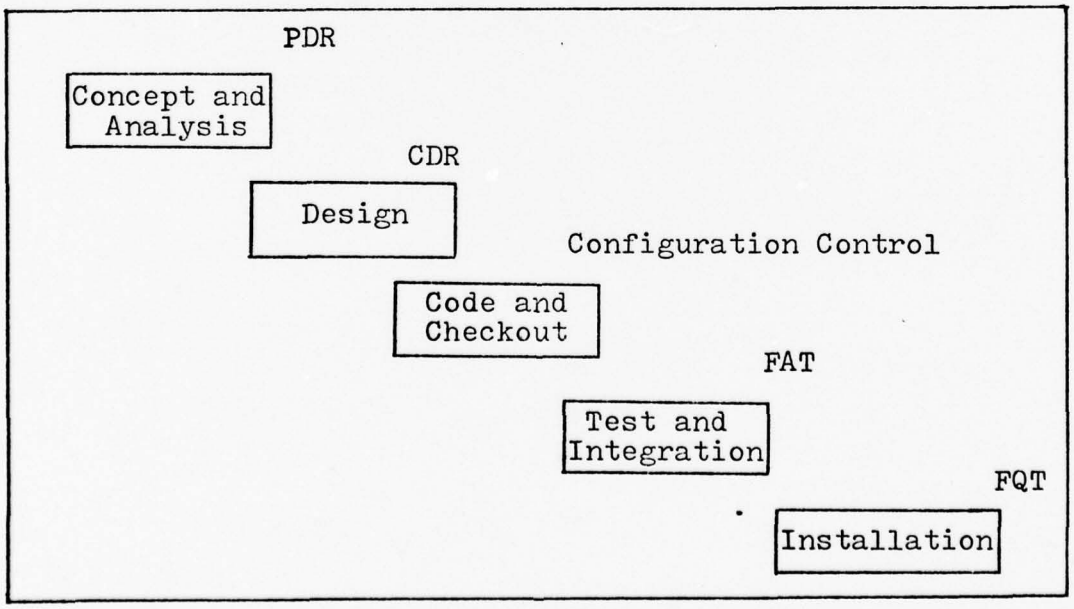


Figure 1. Stages of Software Development Phase

stage encompasses most of the activity from the birth of the idea that led to proposing the software system to the

preliminary design review (PDR) of how the system would be constructed. The design phase refines the preliminary design into the precise module requirements presented in the critical design review (CDR). These requirements are put into the computer code and debugged during the code and checkout stage. The test and integration stage begins when the programmers turn their debugged modules over to the testing group and strict configuration control efforts begin. Test and integration includes inter-module interface testing and culminates in final acceptance tests (FAT) that insure that the software meets the original specifications before entering the installation stage. The final qualification tests (FQT) insure that the software is operating as advertised at the operational sites before entering the operations and support phase of its life cycle.

Operations and Support. The operations and support phase includes four types of activities, shown in Figure 2.

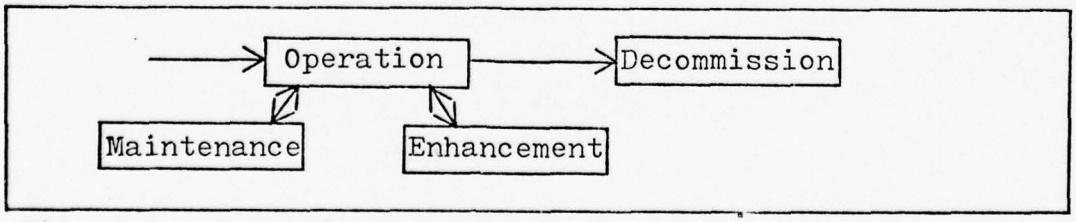


Figure 2. Operations and Support Activities

The most obvious activity is the normal production operation in which the software performs the tasks it was designed to accomplish. The software is obviously not always in an operational state and sometimes requires

maintenance. The error may only cause the system to be in a degraded mode of operation or could bring the system down to a useless state. A third activity of the operation and support phase includes actions taken to enhance the software system either by improving on existing capabilities or adding additional desired capabilities. The final activity of the operation and support phase and of the entire life cycle of the software system is decommissioning. Although this would seem to be a trivial activity, it includes a lot of planning for replacement and backup capability and often a lot of contractual clean-up to bring the life cycle full circle.

Iterative Complexity. Although Figures 1 and 2 depict distinct activities, this is hardly the case in the real world of software systems. Design errors can be discovered well after the critical design review, even as late as the installation stage. Even the boundary between development and operations is not very distinct. Analysis and design play a large role in enhancement activities in particular. Figure 3 shows a much more realistic view of the iterative relationship of the activities in the software life cycle.

Model Life Cycle

Simplifications. Two major simplifications were made to reduce the complexity of the software life cycle process for the purpose of developing this model. The first was to reduce the concept of the life cycle to a smooth continuous function of effort rather than the described

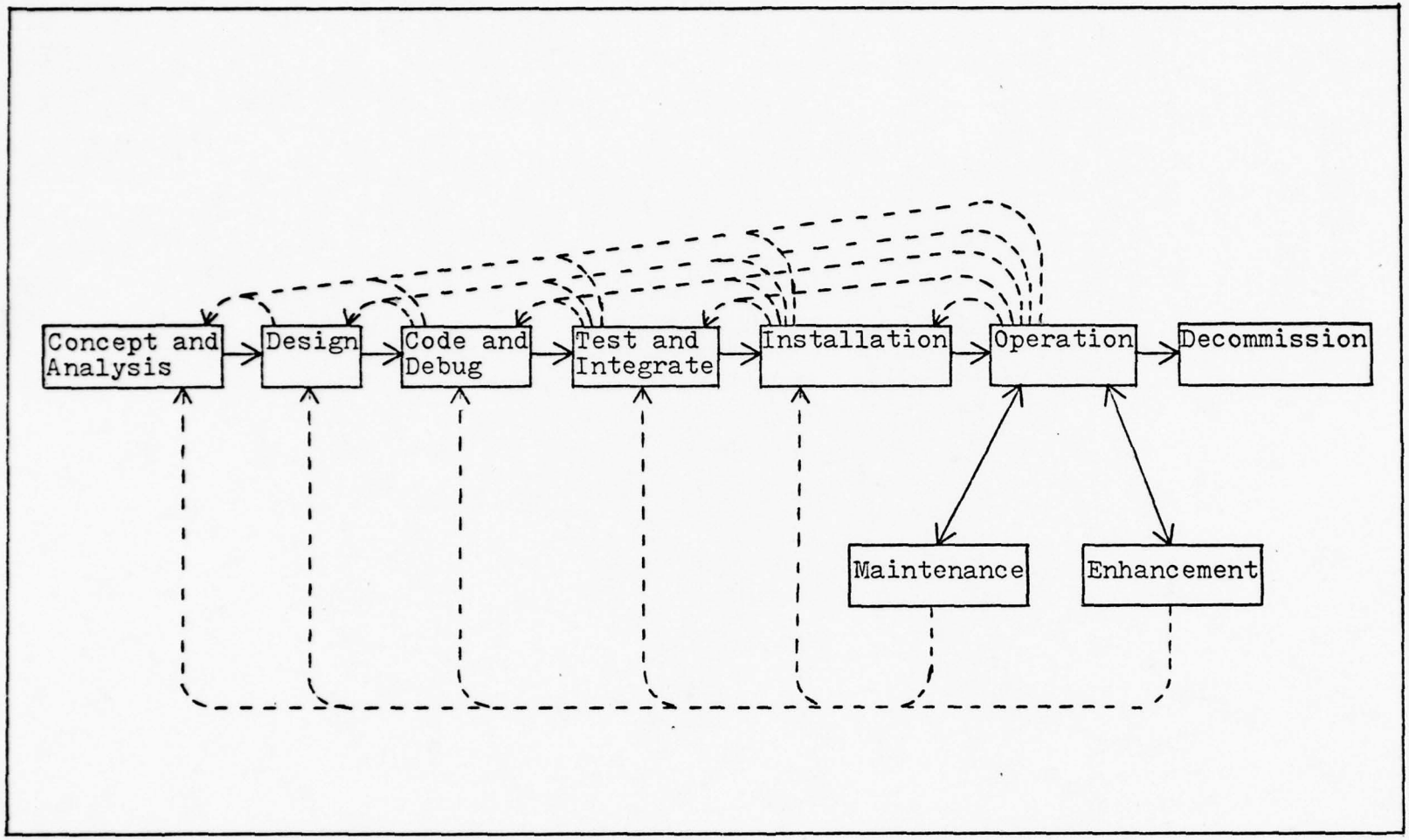


Figure 3. True Life Cycle of Software

distinct activities. After all, what is the value of knowing the cost of the design stage relative to the complexity of breaking down and assigning costs to each stage? The continuous approach allows the model to allocate resources over time rather than to an activity. This would lend itself to the problems of planning and budgeting. A second simplification was to ignore the cost of operations, facilities, administrative support, and hardware since these are somewhat irrelevant to the objectives of this research. The last major simplification was not to consider enhancement activity as a part of the model life cycle. Enhancement of software in truth is the development of a new software system of which a portion already has been completed. As was mentioned earlier, these activities should be costed based on the development of a "new" software system.

Cost Units. Further simplification resulted from the choice of man-months as the unit of cost. Although not a precise unit, the man-month could be statistically standardized. Unlike dollars, inflation has no direct impact on the man-months required to complete a task. However, learning curve effects could have a deflating effect on man-months. Except for the least complex, repetitive operations, the time required to do a task will generally decrease with experience due to learning. Another reason for choosing man-months as a unit of cost is that data for man-months expended on a project would be

rather easy to collect and is easily converted to dollar units with known salary scales.

III Life Cycle Cost Model

Traditional Cost Models

Traditionally, cost models have been restricted to either the development phase or the operation and support phase of a system's life cycle. Even the few models that are called life cycle cost models reduce one phase or the other to a single input of development cost or support cost as an amount or percentage of the other (Refs 5; 6; 7; 8; 9). In the literature search for this research effort, there were no examples of life cycle costing of software systems and, especially, no documented operations and support models. Several modeling techniques are available to perform life cycle costing (Ref 10:15-17).

Unit Cost. The unit cost method is particularly popular in hardware cost models. This method simply adds up the costs of the pieces of the system to derive a system cost. While the cost of a standard gear or audio amplifier may be easily determined, the cost of a search routine or a software fast Fourier transform is not so well known. Unit costing has been applied to software where the size, measured in number of instructions, is multiplied by an average cost per instruction (Ref 10:21). This method is often called decomposition which is the process of breaking the system into ever smaller components until the costs of each of the components can be more accurately estimated.

Analogy. This method relies on the experience of the estimator. If the person or group making the estimate has

had a significant amount of experience with the type of system being developed, then the estimate should be better than if they had less experience. The use of Delphi techniques falls into this category of estimation methods.

Parametric. The parametric method of cost estimation is highly dependent on the availability of reliable data. Parametric modelers attempt to select those parameters of the system which determine the cost and derive a cost estimating relationship. The cost estimating relationship is an equation that sets cost equal to some function of the chosen parameters. The single most effective tool for this method is regression analysis. Typically, in software cost estimation, this function may take the form of

$$c = a I^{b} \qquad \text{(Ref 11; 12; 13) (1)}$$

where c = cost
a = coefficient parameter of the model
I = size of the software system in number of instructions
b = exponent parameter of the model

Parametric models allow the user to enter data on some metric or metrics of the system and mathematically compute an estimate of the cost. The model presented in this thesis is a parametric model.

Life Cycle Cost Function

The parametric model proposed in this thesis is similar in form to the Rayleigh manpower equation. The Rayleigh function as a probability density function has

the form

$$f(t) = \lambda t e^{-\lambda t^2/2} \qquad (2)$$

A form of this function was applied to modeling software development costs by Putnam (Ref 14). The model that he presented took the form

$$y' = 2Kate^{-at^2} \qquad (3)$$

where y' = rate of expenditure in man-years per year
K = total effort in man-years (difficulty)
a = shape parameter (related to type of system)
t = number of years into the development

This function graphically portrays the manpower applied to a software development effort as shown in Figure 4.

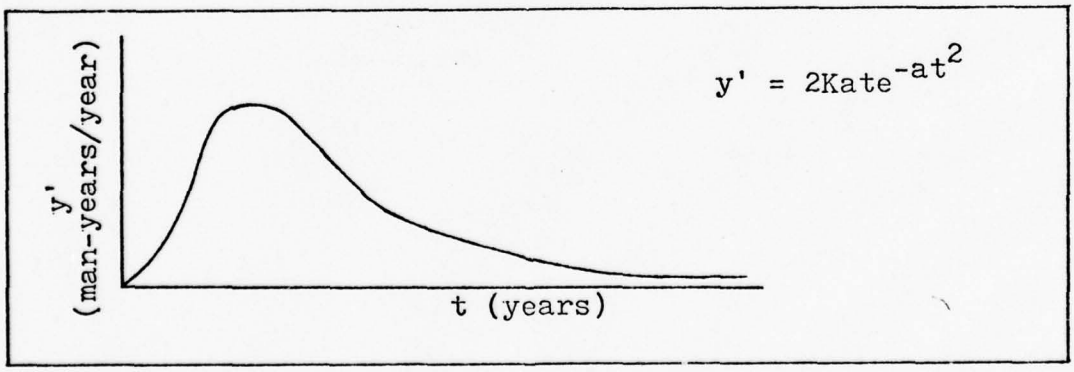


Figure 4. Putnam's Rayleigh Model for Development Costs

Putnam's data was collected while he was assigned to the United States Army Computer Systems Command and primarily concerned with business oriented software systems. Putnam's data was budgetted man-years per year rather than actual man-years per year.

The model proposed in this thesis is very similar to Putnam's except that a third parameter has been added to attempt to model the maintenance tail of the life cycle of the software system. The proposed function has the form

$$m(t) = k_1 te^{-k_2 t^2} + k_3 \qquad (4)$$

where $m(t)$ = manning of effort at time t in man-months per month
k_1, k_2, k_3, = parameters of the function
t = time into the life cycle in months

Several versions of this model were investigated, primarily differring in the form of the last term. Most proved to be somewhat unmanageable mathematically. Equation 4 is easily integrated, differentiated, and otherwise manipulated to allow adequate fitting to data points. This model produces a graph very similar to Putnam's except that it is displaced upward by the constant parameter, k_3 (see Figure 5).

The graph of this function with the proper parameters has the same shape as the budgetted expenditures of effort for software development and maintenance under current policies (see Figure 5). The manning starts out low when the system is undergoing conceptual definition and requirements analysis. The manning grows rapidly as design and coding progress and starts to fall off as the integrated system enters testing. After installation is complete, the manning level for maintenance is pretty close to constant. Each software system is generally

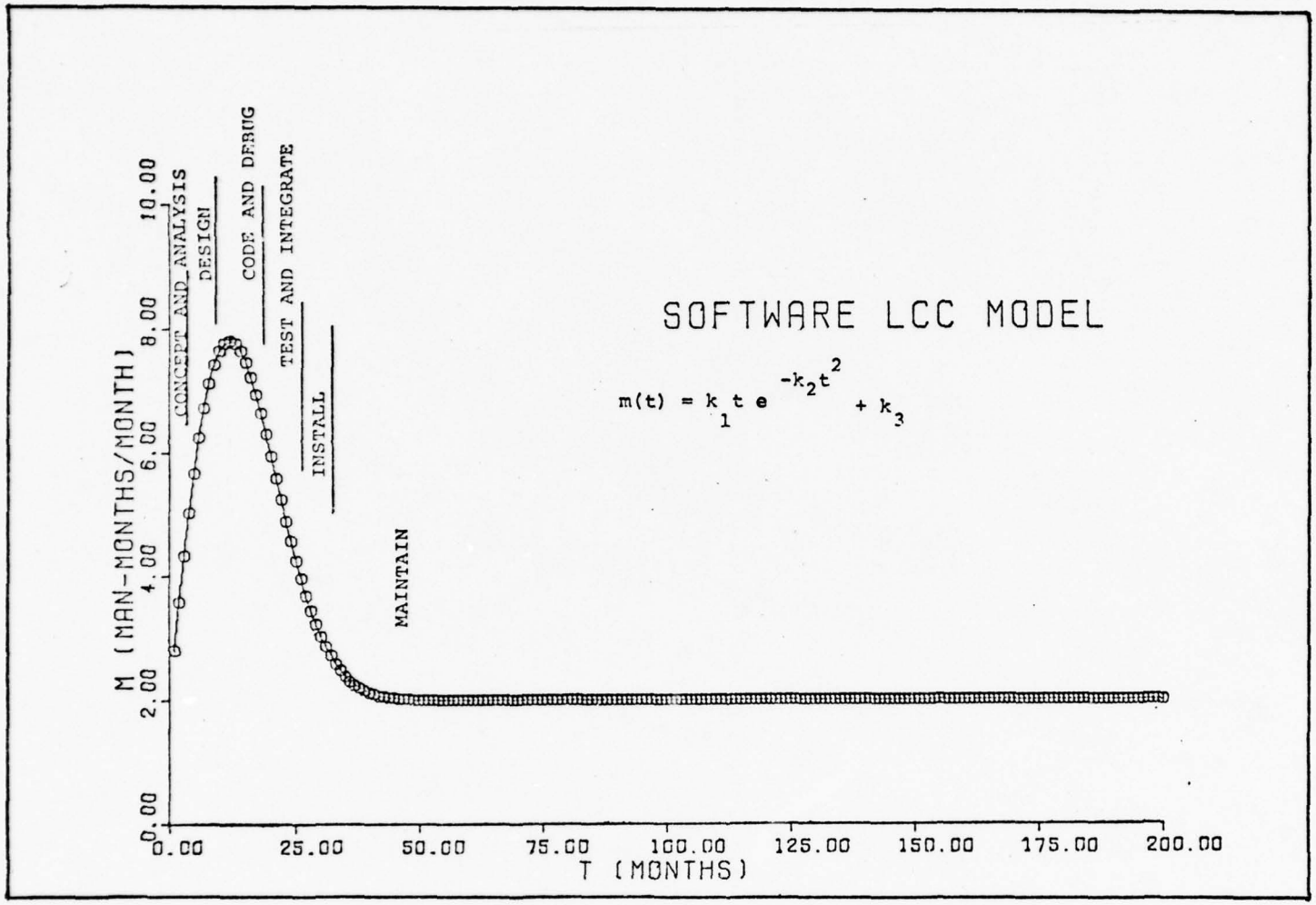


Figure 5. Life Cycle Model Graph

maintained by a distinct group of programmers that are allocated at a nearly constant level over the life cycle of the system.

The Maintenance Tail

To some people it would seem logical that maintenance manpower requirements would decrease over time due to growth in reliability. In other words, as programming and design errors, "bugs", are found and corrected, the time to the next error that makes the system non-operational should increase through the maintenance phase of the life cycle. Any programmer with any experience maintaining software systems can dispute this reliability growth assumption. When errors occur and maintenance action is taken, at least three things can happen. First, the actual error can be truly corrected. The error can be corrected, but the fix includes a new error. Or, a change may be inserted that does not really correct the error that caused the program to be non-operational. So, at best, reliability growth is a probabilistic event depending a lot on the competence of the maintenance programmers.

Likewise, growth in maintainability might be a poor assumption. If maintainability is defined as the time to return a software system to operational status once an error occurs and growth in maintainability is a decrease in the time to correct an error, then growth in maintainability might again seem to be a logical

conclusion. However, several factors might lead another examiner to reach the opposite conclusion, decaying maintainability or increasing time to correct an error. Personnel turnover, common in the software industry not just in the Department of Defense, can inhibit any increase in familiarity with the system. Patchwork fixes can not only introduce new errors, but also complicate the finding of other errors and interface problems. Documentation may decay or simply not exist on new releases. Less than absolute configuration control, especially where multiple versions are in use at separate sites, can easily complicate error identification and correction.

Psychological factors can also control manning levels in maintenance organizations. It may be very difficult to convince an organization responsible for the maintenance of large software systems that it does not need as many people as it originally did to do its job. If the system were split among programmers along functional boundaries, there is indeed an actual dollar cost involved in training the remaining programmers to take over those functions previously maintained by the excess programmer.

One further argument for accepting the compromise constant manning for the maintenance tail resulted from a discussion with the people setting up the software maintenance facility for the F-15 aircraft avionics systems (Ref 15). In determining the number of programmers needed to maintain the F-15's avionics software, they used a

linear model of the form

$$m = \frac{s \times r}{p} \qquad (5)$$

where m = manning requirements
s = size of the software in source statements
r = percent of software recoded per year (.05)
p = programmer productivity (200 instructions per month)

The implication of these assumed values for r and p is that maintenance manning is constant for a given system size and that

$$m = \frac{s}{48000} \qquad (6)$$

or that one programmer is required for every 48,000 lines of source code. This suggests that there might be an upper limit on the size of software that a single maintenance programmer can keep up with adequately. The value of this limit might be determined by factors such as programmer experience or the modern programming practices used to produce the software, structured design or modularization.

The only way to verify the concept of the constant maintenance tail is to collect long term data on actual systems and compare the data to the model. Assuming that the data exists, the next pertinent question would be how to build the model from the data.

Parameter Evaluation

In order to evaluate the parameters of the model, k_1,

k_2, and k_3, the model function is fit to the data available. Several methods were available to do the fitting. All of the methods started with an initial evaluation of the maintenance tail parameter, because its value is in fact the asymptote of the curve of the function as is obvious from Equation 7 below. In

$$\lim_{t\to\infty} m(t) = \lim_{t\to\infty} k_1 t e^{-k_2 t^2} + k_3 = k_3 \qquad (7)$$

consonance with the arguments presented in the previous section, k_3 is calculated from the data by averaging all the values for manning after the software was delivered, t_d,

$$k_3 = \frac{\sum_{i=d+1}^{n} m(t_i)}{n - d} \qquad (8)$$

where n = number of data points available
d = number of months in development prior to delivery
$m(t_i)$ = actual manning during month t_i

The location parameter, k_2, and the shape parameter, k_1, can now be determined by an experimental, trial and error technique. The technique required a lot of computing and plotting of functions on top of actual data while adjusting the two parameters, k_1 and k_2, until a reasonable visual fit was found. In order to expedite the

process, a computer program, LCCPLOT, was written and is included as Appendix B. A more reasonable approach would be to directly compute estimates of k_1 and k_2 from the available data.

The method of linear least squares offers one means of computing the parameters. To apply this technique, the function must be linearized. This can be done by moving k_3 to the left side of the equation and taking the natural logarithm of both sides resulting in the following:

$$\ln (m(t) - k_3) = \ln (k_1) + \ln (t) - k_2 t^2 \qquad (9)$$

which is of the linear form

$$y = b_0 + b_1 g_1(t) + b_2 g_2(t) \qquad (10)$$

One major drawback of this method is that should k_3 ever be equal to m(t), then the natural logarithm is undefined. Applying this method resulted in differences between total cost in man-months from actual data and the model function of as much as ten percent for the data available. So, another more appropriate method was sought.

By using the derivative to solve for the maximum of the model function, the parameter, k_2, can be determined. The derivative evaluated at the time of maximum manning, t_{max}, must be equal to 0 as shown here:

$$\left.\frac{d\,m(t)}{dt}\right|_{t_{max}} = k_1 e^{-k_2 t_{max}^2} - 2k_1 k_2 t_{max}^2 e^{-k_2 t_{max}^2} = 0 \quad (11)$$

$$\text{or} \quad k_2 = \frac{1}{2t_{max}^2} \quad (12)$$

Since t_{max} can be fixed through inspection of the data, k_2 is determined. In the application presented later, t_{max} is the time at which the maximum manning level first occurs in the data. It is also apparent that k_2 is a location parameter determining the location of the peak of the model function.

Assuming that k_2 and k_3 have been determined, k_1 can be determined by thinking in terms of the purpose of the model, that is, to accurately model the life cycle cost of the system. But, the life cycle cost of the system is simply the integral of m(t) over the life cycle of the system or

$$M(t_{lc}) = \int_0^{t_{lc}} m(t)dt = \frac{k_1}{2k_2}\left(1 - e^{-k_2 t_{lc}^2}\right) + k_3 t_{lc} \quad (13)$$
$$= LCC$$

where M(t) = accumulated cost at time t in man-months
t_{lc} = number of months in the life cycle
LCC = total life cycle cost in man-months

When working with the data available, t_{lc} is replaced by t_a where t_a is the number of months for which data are

available. $M(t_a)$ is easily computed from the actual data by the following:

$$M(t_a) = \sum_{i=1}^{a} m(t_i) \tag{14}$$

And now, manipulating Equation 13 yields

$$k_1 = \frac{2k_2 \, (M(t_a) - k_3 t_a)}{1 - e^{-k_2 t_a^2}} \tag{15}$$

which determines k_1 in terms of known data.

Using Equations 8, 12, and 15, the parameters of the model function can be reduced to numbers and the function plotted against the actual data. This capability is also provided via the computer program, LCCPLOT (see Appendix B). This method makes computing the parameters of the life cycle cost function straightforward when data is available, but how are the parameters determined for unknown software systems?

Factor Identification

One of the objectives of this thesis was to relate modern programming practices and management decisions to their effects on the life cycle cost of software systems. An appealing assumption is made concerning these factors and the parameters of the life cycle cost model proposed here. That assumption is that the parameters of the model

can be expressed as functions of a set of known factors of the software system as shown here:

$$k_1 = f(\text{known factors}, f_i) \tag{16}$$

$$k_2 = g(\text{known factors}, f_i) \tag{17}$$

$$k_3 = h(\text{known factors}, f_i) \tag{18}$$

One further simplifying assumption is that the functions, f, g, and h, are linear combinations of the known factors. That is, the functions are of the form

$$k_1 = \sum_{i=1}^{n} a_i f_i \tag{19}$$

$$k_2 = \sum_{i=1}^{n} b_i f_i \tag{20}$$

$$k_3 = \sum_{i=1}^{n} c_i f_i \tag{21}$$

Very few references consider factors such as size of the software to be linearly related to cost, however, linearity is assumed here for simplicity.

The factors chosen for analysis must meet certain criteria. The first and most practical criteria is availability. The factor must be known and reported along with the manning data on the system. The factors must be quantifiable. Logical ones and zeros can be used to indicate the use or non-use of given practices. Percentages,

counts, and measures are more obviously quantifiable forms of reporting factor values. As mentioned in the introduction, consistency of definitions and measurement units should also be a strict criteria. One other criteria is variability. Unless the factor varies from one system to the next, there is no point in asking questions about that factor. If all the data comes from systems written in the COBOL language, then making the language a factor is a useless effort.

Some of the factors that can affect the cost of software that were reported and found in the literature search have been compiled and regrouped in Figure 6. Given sufficient data on life cycle cost and factors for each of the systems, the resultant systems of linear equations can be solved to reveal the coefficients of the model, a_i, b_i, and c_i. The linear equations can be expressed in matrix form as

$$\begin{bmatrix} k_{11} \\ k_{12} \\ k_{13} \\ \vdots \\ k_{1m} \end{bmatrix} = \begin{bmatrix} f_{11} & f_{21} & f_{31} & \cdots & f_{n1} \\ f_{12} & f_{22} & f_{32} & \cdots & f_{n2} \\ f_{13} & f_{23} & f_{33} & \cdots & f_{n3} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ f_{1m} & f_{2m} & f_{3m} & \cdots & f_{nm} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} \qquad (22)$$

where k_{1j} = parameter k_1 evaluated for system j
m = number of data sets available
f_{ij} = value of factor i for system j

n = number of factors used in the model
a_i = coefficients of the linear function relating the parameter, k_1, to the factors used

Similar matrix equations can be written for the other parameters, k_2 and k_3, where b_i and c_i would replace the a_i as the coefficients of the linear function.

Requirements	Management
changes (ECP, SCN)	configuration management
reliability	librarian
maintainability	chief programmer teams
customer experience	type of contract
documentation	testing (automated, IV&V)
	standardization
Hardware	Software
memory constraint	application
time (CPU) constraint	language
first-time development	support software/tools
concurrent development	structured design/coding
availability	modularization
turnaround/throughput	walkthroughs
Sizing	People
total source	total number
total object	mixture (technical/admin)
delivered source/object	experience level
newly developed source	education level
data elements and types	turnover rate
number of reports	
number/size of files	

Figure 6. Factors That May Affect Life Cycle Cost (Refs 12:15-16; 22(Vol III):5-7; 23:3-1-69; 14:42; 16:7; 17:27-32)

The next step is, of course, the solution of this set of matrix equations for the coefficients so that the model can be used to predict the life cycle cost of a system from only the factors used in the model. In other words, the goal is to compute the function parameters, k_1, k_2, and k_3, for a non-existent system using estimates or known values for the factors used in the matrix equations. If the number of factors, n, to be used in the model is greater than the number of data sets available, m, then the equations can not be solved for the coefficients. If n is equal to m, then there is a solution to the matrix equations which can be found using Cramer's Rule assuming that the factor matrix is non-zero. When more data sets are available than factors that are included in the model (m greater than n), then the method of least squares can be used to find the best (by least squares standards) fit.

To make these types of calculations, the digital computer is an excellent tool. Appendix C describes a program, LCCMODL, that was written to implement the model presented in this thesis. It accepts the data sets (manning and factors), computes fitted parameters, solves the matrix equations and makes predictions of function parameters using the derived coefficients and given factor values.

Sensitivity Analysis

The purpose of performing sensitivity analyses is to

gain insight as to which variables in a problem will, when varied, have the greatest effect on the final result. With respect to the model presented in this thesis, sensitivity analysis is used to identify the effects of the function parameters and the system factors on the life cycle cost of the software system.

Parameters. Two methods can be applied to measure the sensitivity of life cycle cost to changes in the values of the function parameters. The first, a manual method, is to simply vary the value of one parameter while holding the others constant and observe the resultant change in cost. Table I shows a sample manual sensitivity analysis assuming a life cycle of 200 months. Figure 7 then shows the manning curves which display the time phased effects of variations in the parameters on cost.

Table I

Manual Parameter Sensitivity Analysis

Parameter Values			Life Cycle Cost
k_1	k_2	k_3	M(t=200)
2.0	.0002	2.0	5398.33
1.9	.0002	2.0	5148.41
2.1	.0002	2.0	5648.24
2.0	.0001	2.0	5308.42
2.0	.0003	2.0	5399.97
2.0	.0002	1.9	5378.33
2.0	.0002	2.1	5418.33

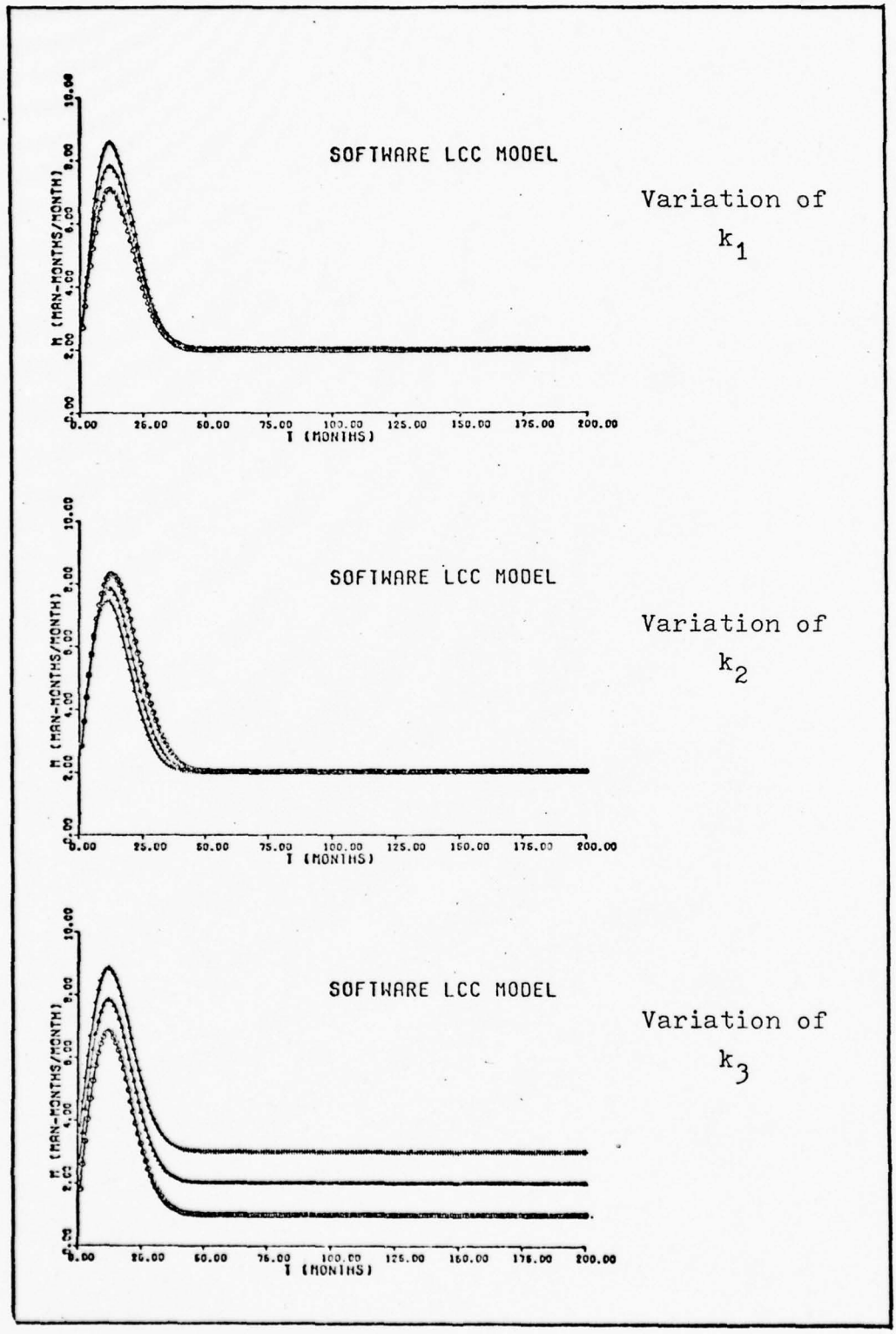


Figure 7. Sample Sensitivity Analysis

The method of partial derivatives provides more information about the sensitivity of the life cycle cost over the life cycle of the system. For example, the sensitivity of M(t) with respect to the parameter, k_1, is simply the partial derivative of M(t) with respect to k_1 or

$$\frac{\partial M(t)}{\partial k_1} = \frac{1 - e^{-k_2 t^2}}{2k_2} > 0 \qquad (23)$$

which is greater than zero for all positive values of k_2 and t. This implies that any increase in the parameter value will produce an increase in the life cycle cost of the system. Similarly, the partial derivative of M(t) with respect to k_3 is

$$\frac{\partial M(t)}{\partial k_3} = t > 0 \qquad (24)$$

which also is greater than zero since t is always greater than zero. The partial derivative of M(t) with respect to k_2 is not nearly so trivial to evaluate:

$$\frac{\partial M(t)}{\partial k_3} = -\frac{k_1}{2k_2^2}\left(1 - e^{-k_2 t^2}\right) + \frac{k_1}{2k_2}\left(t^2 e^{-k_2 t^2}\right) \qquad (25)$$

Although the second expression is obviously positive when k_1, k_2, and t are positive, the first term is negative.

However, for significantly large t, the second term will be dominated by the first and make the value of the partial less than zero. And, any increase in k_2 will cause a decrease in the total life cycle cost. It is interesting to note that this parameter can result in more expenditure of resources during development (low t) and less expenditure during the maintenance tail (high t).

Factors. From this information about the parameters of the function, it is possible to draw direct conclusions about the sensitivity of life cycle cost to the values of the factors chosen for the model. It follows from the linear relation of the factors to the parameters that

$$\frac{\partial k_1}{\partial f_i} = a_i \tag{26}$$

$$\frac{\partial k_2}{\partial f_i} = b_i \tag{27}$$

$$\frac{\partial k_3}{\partial f_i} = c_i \tag{28}$$

The sign of the coefficients will indicate the direction of the effect on life cycle cost, positive implying an increase and negative implying a decrease in cost when the factor value is increased. To determine the relative magnitude of the effects of each factor, the partial derivatives must be normalized by dividing by the value of the parameter or

$$\frac{\frac{\partial k}{\partial f_i}}{\frac{}{k}} = \frac{a_i}{k} \tag{29}$$

Going back to the sensitivity of life cycle cost with respect to the parameter, k, the normalized sensitivity of life cycle cost with respect to the factor, f_i, due to its affect on the parameter k is given by

$$\frac{\frac{\partial M(t)}{\partial f_i}}{M(t)} = \frac{\frac{\partial M(t)}{\partial k} \cdot \frac{\partial k}{\partial f_i}}{M(t)} \tag{30}$$

In particular, the normalized sensitivity of life cycle cost to the factor, f_1, due to its affect on parameter k_1 is

$$s_{k_1 f_1} = \frac{\frac{1 - e^{-k_2 t_{lc}^2}}{2k_2} \cdot a_1}{M(t_{lc})} \tag{31}$$

Note here that $s_{k_1 f_1}$ does not indicate the total sensitivity of cost to factor, f_1. It does not include the affects of factor, f_1, on cost due to its affects on the other parameters, k_2 and k_3. From these computations, it can be determined which of the factors in a model have the greatest effect on the total life cycle cost.

Selection. It is important to have a tool to

determine which factors should be included in the model. Sensitivity analysis provides this tool. The procedure is to build the model with the available data and a first set of desired factors. A sensitivity analysis of the resulting model parameters and coefficients can reveal which factor is least influential in determining life cycle cost. This factor can be replaced by another candidate factor in a second solution of the model. This procedure can then be continued until the sensitivity analysis shows the best distribution of contribution to the life cycle cost. Those factors that most dominate the life cycle cost should be reduced in effect while those that contributed the least should increase in importance. Ideally, the magnitudes of contribution of each of the chosen factors would be equal. In order to properly demonstrate this process, an example is presented in the next chapter with a formal computational algorithm.

IV Sample Application

Sample Data

The data used for this sample application of the modeling algorithm is taken from a technical report from Sperry-Univac Defense Systems (Ref 23). The raw manning data and factors for each of four software systems is displayed in Appendix A. The remainder of this chapter is a step-by-step example of the computational algorithm that will implement the proposed model.

Computational Algorithm

Step 1. The first step of the algorithm is to fit a set of parameters, k_1, k_2, and k_3, to each data set. Equations 8, 12, and 15 are implemented in the computer program, LCCMODL, described in Appendix C. Because the data available for this example is for development only, special arrangements have to be made for the fitting of k_3. When the input to the program that indicates the number of development data points is the same as the total number of data points, the program expects to read in the assumed value of k_3 from the data deck. For the current example these values were computed by dividing the software size in total source instructions by 48,000 as suggested by the F-15 maintenance example. The results were left as fractions rather than rounded up to the next whole man-month. The resultant parameters are shown in Table II.

Table II

Fitted Parameters

Program	k_1	k_2	k_3
1	.64969	.000520	1.875
2	2.24267	.000297	10.417
3	2.20299	.002551	.554
4	.79777	.003472	.274

Step 2. The second step of the algorithm is to select the factors to be used in the first attempt at the model. In order to verify the usefulness of the model only the first three data sets will be used to build the model. This means that only three factors can be included in the model ($n \leq m$, see Chapter III). The following three factors were chosen at random for the first round:

1. percent HOL (higher order language)
2. programmer qualification (combines education and experience)
3. development on target hardware

Step 3. The next step is to take the fitted parameters and known factors and solve the matrix Equations 21 for the model coefficients, a_i, b_i and c_i (i=1,2,3). This calculation is also available in the computer program, LCCMODL, as a call on a subroutine that solves the system of linear equations. The results of this first attempt at the model are shown here:

$$k_1 = 2.649\ (\%\ \text{HOL}) + .013\ (\text{PROG QUAL}) - .853\ (\text{DEV ON TGT}) \quad (32)$$

$$k_2 = .000233\ (\%\ \text{HOL}) + .000043\ (\text{PROG QUAL}) - .001052\ (\text{DEV ON TGT}) \quad (33)$$

$$k_3 = 13.6714\ (\%\ \text{HOL}) - .1066\ (\text{PROG QUAL}) + .8356\ (\text{DEV ON TGT}) \quad (34)$$

A quick check on the model is to now use the coefficients to predict the values of the parameters to fit the fourth set of data. The results of the prediction are k_1 = 3.6975, k_2 = .003276, and k_3 = 4.8910, not very close to the values presented in Table II.

Step 4. The fourth step of the algorithm is to evaluate the relative contribution of each factor to the life cycle cost by means of sensitivity analysis. The computer program LCCMODL does these computations using Equation 30 (see Appendix C). The appendix includes the results for this set of factors as a sample. Table III shows the results of these calculations for the first set of data. Factor 2, programmer qualification, turns out to

Table III

Sensitivity Analysis 1 *

Parameter k	Factors 1	2	3
k_1	2.548	.012	- .820
k_2	.279	- .051	1.263
k_3	2.736	- .021	.167

* assumes life cycle of 200 months

have the least significant effect on life cycle cost. The procedure is then to select another factor to take its place and reaccomplish the sensitivity analysis. The factor chosen to replace programmer qualification is whether or not modular design was employed to develop the software system. The results of the sensitivity analysis of this combination for the first set of data are shown in Table IV. This process can be repeated over and over until

Table IV

Sensitivity Analysis 2

Parameter k	Factors 1	2	3
k_1	2.50966	- .328914	.788334
k_2	.43868	- .791454	- 3.29572
k_3	2.80248	- .689697	- 1.37444

the modeler is satisfied that he has an adequate model of the data at hand, perhaps testing the model by allowing it to predict the parameters of another data set not used to build the model. The results here for the fourth data set are k_1 = 3.42960, k_2 = .00237932, and k_3 = 7.13554. Again, the predicted values are not very close to the fitted parameters in Table II. In fact, the deviation is greater.

Step 5. The final step in the computational algorithm is really the final objective of the process. Assuming that the model has been built and verified by checking

against known data, it can be used to predict the necessary manning requirements for unknown systems. It must be assumed that the factors used in building the model can be reasonably estimated when the system does not yet exist. Estimating the factors, such as size, can be nearly as difficult a task as estimating the cost was before the model was applied.

Summary. As a summary, Figure 8 lists the five steps of the computational algorithm.

Step	Action
1	fit parameters to data
2	select factors to include
3	solve for model coefficients
4	perform sensitivity analysis
5	predict parameters of unknown system

Figure 8. Computational Algorithm

Conclusions

The calculations shown in this chapter are only intended to act as a sample format of how the computational algorithm is to be applied. There are many variations in combinations of factors that could be tested. It is clear from the inability of the two iterations shown here to accurately predict the parameters of the fourth data set that the model needs more refinement. Whether this is the result of an inadequate model, inadequate data, or poor

choice of factors is still a matter of conjecture. The two iterations presented here represent the best results of about a dozen trials judged by predictive capability.

V Management Applications

The model presented in this thesis could provide a direct means to evaluate the impact of current and future programming or management practices on the life cycle cost of systems. The computational algorithm for building the model and the computer programs presented provide the manager some significant tools. Of course, any use of this model should be preceded by adequate data collection, model validation, and, if necessary, significant modification.

Pre-Contract Applications

Preliminary life cycle costing of software system proposals before entering development, while not very accurate, can be used in evaluating proposals. The request for proposal should specifically call for the values of the factors to be used in the model. The factors must be very specifically defined and quantified in the same way on each proposal.

In a more general sense, the model has by indicating whether the affect of a given factor increases or decreases life cycle cost shown how the factors should be managed. Factors that increase the life cycle cost should be avoided or at least controlled. Factors that help to reduce life cycle cost should be encouraged or required contractually or by regulation. This information could be available from the construction of the model before a contract was awarded.

Development Applications

After the contract is awarded, a software development manager can continue to use the derived model as a planning and checking tool. Many studies have been performed to investigate the percentage of development effort involved in the five stages of the software development phase (see Table V). Using these percentages, or others generated

Table V

Proportion of Development Effort by Stage

Stage	Reference - number and page					
	18:B-3	12:28	14:171	14:171	14:171	Ave
Concept & Analysis	.05	.05	.20	.20	.20	.140
Design	.25	.31	.16	.20	.20	.224
Code & Debug	.35	.23	.16	.25	.24	.246
Test & Integration	.25	.19	.27	.20	.20	.222
Installation	.10	.22	.21	.15	.16	.168

for his particular case, the manager could check his expenditures of manpower. The time to the end of each stage can be calculated as that time, t, at which

$$M(t) = p \cdot M(t_d) \tag{35}$$

where p = percentage of development effort for stage
$M(t_d)$ = total development cost
d = number of months in development

In addition, a plot of the model manning curve and the actual expenditure of manpower can be compared with limits set to trigger an in-depth investigation of gross deviations

from the planned course.

Maintenance Applications

The comparison of actual and predicted manpower needs during the maintenance phase of the life cycle would be one important application of this model. Because the lack of data led to the flat maintenance tail assumption, there is little that can be done to increase the value of the model other than to collect data to verify or modify the tail of the model. In all stages of the life cycle of a system, data should be compared to the predicted manning levels. When deviations occur, any factors that might explain the deviation should be reported and perhaps applied in a re-building of the model. A centralized point for software data collection would be invaluable.

VI Recommendations

Data

Collection. By far the most important recommendation to be made is that data on the software life cycle costs be collected and made available to investigations such as this. This is hardly a new recommendation, but it bears repeating for emphasis (Refs 19:92; 20:3-2). Without data to test, estimate parameters, verify, and validate the model and assumptions, the parametric approach really provides no extra insight to help control software costs. Maintenance data was found to be especially lacking during this effort.

It is important here to note the difference between budgetted and actual costs even in man-months. Because three people are assigned to maintain a software system does not mean that it actually took three man-months of effort to maintain the system during a given month. Considering the growing trend to in-house maintenance of software systems in regional centers, it may well come to pass that maintenance programmers could be shared between systems, i.e., that maintenance effort may be budgetted in fractions rather than whole numbers of man-months. The sharing could be done on the basis of areas of expertise such as search techniques, statistical applications, or specialized interfaces. For instance, an expert in data filtering could be shared by several systems. The only obvious solution to the lack of data problem is to require that data be collected and made available by regulation

for in-house projects and as a deliverable item for contracted work.

Standardization. If cost reporting is to be required, the reporters must be fully aware of exactly what they are to report. In other words, standards must be established. Definitions of terms must be specific. Structured programming must mean the same thing to all reporters of data. Units of measurement must be consistent from project to project. Efforts have been made as early as 1966 (Ref 21) to specify cost reporting elements and were repeated in 1976 (Ref 13). More recently, 1976 to 1977, a major study was commissioned by the Rome Air Development Center to generate recommendations for a data collection system to establish a repository for data to be used in studying productivity, reliability, and cost of software (Ref 22). This represents at least one step in the right direction.

Follow-on Modeling

Every researcher, at least subconsciously, wants his work to be used and continued. In the case of this modeling approach, there are indeed a lot of things left to be done. Before accepting or rejecting the model, it needs to be given adequate testing as data becomes available. Should the data indicate that the model provides inadequate results for accurate prediction, several modifications can be made. One modification would be to change the form of the model function with either more or fewer parameters. Another more challenging route might be

to explore the effects of non-linear factors. These modifications and more yet to be conceived of could lead to a useful life cycle costing tool for the production and maintenance of software systems.

Other Applications

Although there are those in the software business that still insist that programming is an art and bears no resemblance to any field of hardware engineering, there are others that believe software engineering people have wasted considerable valuable time in re-inventing the software wheels. Most everyone will concede that there are differences between computer programs and radars, but do these differences conceptually amount to more than the differences between radars and automobiles? While working with this modeling approach, I was struck by the similarity of the manning curve for software systems and that I conceived would be applicable to hardware systems. With a different set of factors, could the same approach be used to model hardware systems or, for that matter, any research and development effort?

Bibliography

1. Riley, M. J. "Implementing Life Cycle Cost: A Financial Manager's Role." Research Study. Maxwell AFB, Alabama: Air Command and Staff College (AU), May 1977. (ADB018360L)

2. "Software Improvement Plan Pushed," Aviation Week & Space Technology, 104: 41-43 (5April 1976).

3. "USAF Pressing Reduced Software Costs," Aviation Week & Space Technology, 101: 63-4 (29 July 1974).

4. AFR 800-14. "Management of Computer Resources in Systems." Washington, D.C.: United States Air Force, 12 September 1975.

5. Baron, D. A. and R. E. Mortenson. "A Logistics Life Cycle Support Cost Model." Master's Thesis. Wright-Patterson AFB, Ohio: Air Force Institute of Technology (AU), August 1969. (AD863842)

6. Brannon, R. G. "Army Life Cycle Cost Model - Volume I: User's Guide." Final Report, DCA-R-15. Washington, D.C.: Directorate of Cost Analysis, January 1976. (ADA021900)

7. Collins, D. E. "Analysis of Available Life Cycle Cost Models and Actions Required to Increase Future Model Applications." Technical Report, ASD-TR-75-25. Wright-Patterson AFB, Ohio: Joint AFSC/AFLC Commanders' Working Group on Life Cycle Cost, June 1975. (ADA014772)

8. Hamilton, J. L. "Life Cycle Cost Modeling." Technical Report, TR-68-8. Washington, D.C.: United States Army Materiel Command, December 1968. (AD684335)

9. Stone, H. S. "Life Cycle Cost Analysis of Instruction-set Architecture Standardization for Military Computer-based Systems." Final Report (draft). United States Research Office, January 1978.

10. Clapp, J. A. "A Review of Software Cost Estimation Methods." Technical Report, ESD-TR-76-271. Bedford, Massachusetts: The Mitre Corporation, August, 1976. (ADA029748)

11. Herd, J. H., J. N. Postak, W. E. Russell, and K. R. Stewart. "Software Cost Estimation Study- Volume I: Study Results." Technical Report, RADC-TR-77-220. Rockville, Maryland: Doty Associates, Inc., June 1977. (ADA042264)

12. Doty, D. L., P. J. Nelson, and K. R. Stewart. "Software Cost Estimation Study - Volume II: Guidelines

for Improved Software Cost Estimating." Technical Report, RADC-TR-77-220. Rockville, Maryland: Doty Associates, Inc., August 1977. (ADA044609)

13. Graver, C. A., E. E. Balkovich, W. M. Carriere, and R. Thibodeau. "Cost Reporting Elements and Activity Cost Tradeoffs for Defense Systems Software - Volume I: Study Results." Final Report, CR-1-721. Santa Barbara, California: General Research Corporation, November 1976.

14. Putnam, L. H. and R. W. Wolverton. Tutorial - Quantitative Management: Software Cost Estimating. New York: Institute of Electrical and Electronics Engineers, Inc., 1977.

15. Sherrill, C. Warner-Robins ALC/MMECV (F-15 ASIF). Telephone Interview, July 1978.

16. Farr, L. and B. Nanus. "Factors that Affect the Cost of Computer Programming." Technical Memorandum, ESD-TM-1447/000/02. Santa Monica, California: System Development Corporation, June 1964. (AD447329)

17. Farr, L. and H. J. Zagorski. "Factors that Affect the Cost of Computer Programming - Volume II: A Quantitative Analysis." Technical Documentary Report, ESD-TDR-64-448. Santa Monica, California: System Development Corporation, September 1964. (AD607546)

18. Black, R. K. F., R. Katz, M. D. Gray, and R. P. Curnow. "BCS Software Production Data." Technical Report, RADC-TR-77-116. Seattle, Washington: Boeing Computer Services, Inc., March 1977. (ADA039852)

19. "Findings and Recommendations of the Joint Logistics Commanders' Software Reliability Working Group (SRWG Report) - Volume I: Executive Summary." Technical Report, HQ-AFSC-TR-75-05. November 1975. (ADA018881)

20. Craig, C. R., et al. "Software Reliability Study." Technical Report, RADC-TR-74-250. Redondo Beach, California: TRW Systems Group, October 1974. (AD787784)

21. Weinwurm, G. F. "Data Elements for a Cost Reporting System for Computer Program Development." Technical Report, ESD-TR-66-411. Santa Monica, California: System Development Corporation, August 1966. (AD637804)

22. Willmorth, N. E., M. C. Finfer, and M. P. Templeton. "Software Data Collection Study." Technical Report, RADC-TR-76-329. Santa Monica, California: System Development Corporation, December 1976,
"Volume I: Summary and Conclusions" (ADA036115)

"Volume II: An Analysis of Software Data Collection Problems and Current Capabilities" (ADA036116)
"Volume III: Data Requirements for Productivity and Reliability Studies" (ADA036064)
"Volume IV: Data Management System Interface" (ADA036065)
"Volume V: Survey of Project Managers" (ADA036066)
"Volume VI: Proceedings of the Data Collection Problem Conference" (ADA037701)
"Volume VII: Compendium of Procedures and Parameters" (ADA036247)

23. Branning, W. E., D. M. Willson, J. P. Schaenzer, and W. A. Erickson. "Modern Programming Practices Study Report." Technical Report, RADC-TR-77-106. Saint Paul, Minnesota: Sperry-Univac Defense Systems, April 1977. (ADA040049)

24. "CDC 6600 CALCOMP Plotter Manual." Publication CC05. Wright-Patterson AFB, Ohio: Aeronautical Systems Division Computer Center (AFSC), February 1973.

A Sample Data

The data used in the sample calculations of Chapter IV was provided by Sperry-Univac Defense Systems in a Rome Air Development Center sponsored technical report (Ref 23: 1-31). The data tabulated in Table A-I is the manning data for the four software systems reported in the report.

Table A-I

Sperry-Univac Manning Data

Month	Program				Month	Program				Month	Program		
	1	2	3	4		1	2	3	4		1	2	3
1	5	10	5	2	31	17	66	3	-	61	7	-	-
2	5	13	8	2	32	17	67	3	-	62	7	-	-
3	5	13	8	3	33	17	67	3	-	63	7	-	-
4	5	15	10	3	34	17	71	3	-	64	7	-	-
5	5	15	12	4	35	17	72	-	-	65	7	-	-
6	5	15	14	4	36	17	72	-	-	66	7	-	-
7	5	25	16	4	37	17	73	-	-	67	7	-	-
8	5	25	19	4	38	17	73	-	-	68	3	-	-
9	5	25	20	4	39	17	73	-	-	69	3	-	-
10	5	34	21	4	40	13	73	-	-	70	2	-	-
11	5	34	22	4	41	8	74	-	-	71	2	-	-
12	5	34	22	5	42	8	74	-	-	72	3	-	-
13	10	40	22	5	43	8	74	-	-	73	3	-	-
14	10	40	23	5	44	8	74	-	-	74	7	-	-
15	10	40	22	5	45	8	74	-	-	75	7	-	-
16	10	45	22	5	46	8	74	-	-	76	8	-	-
17	15	45	22	5	47	8	73	-	-	77	8	-	-
18	15	45	19	5	48	8	73	-	-	78	7	-	-
19	15	49	17	5	49	8	73	-	-	79	7	-	-
20	15	49	14	5	50	8	73	-	-	80	3	-	-
21	15	49	13	5	51	8	73	-	-	81	-	-	-
22	15	52	12	5	52	8	55	-	-				
23	15	53	11	5	53	8	55	-	-				
24	15	53	10	5	54	8	55	-	-				
25	15	56	8	5	55	8	35	-	-				
26	16	56	7	5	56	8	35	-	-				
27	16	56	6	1	57	8	35	-	-				
28	16	60	4	1	58	8	35	-	-				
29	16	60	4	-	59	8	-	-	-				
30	16	60	3	-	60	7	-	-	-				

The units are man-months per month. Table A-II shows the factor data available on the four systems also found in the technical report.

Table A-II

Sperry-Univac Factor Data

Factor	Program 1	2	3	4
Size in delivered source	90000	500000	26600	13150
Real-time application	1	1	1	1
Top-down structured design	0	0	1	1
Structured coding	0	0	0	1
Memory constraint	.50	.50	.52	.50
Percent HOL used	38	99	53	100
Programmer qualification education and training	39.0	37.1	62.8	82.4
Developed on target machine	1	1	0	0
Pages of documentation	8059	27014	3507	2259
Command and control application	1	1	1	1
Modular design	0	0	1	1
Program librarian	1	0	1	1
Structured narative	1	0	0	1
Flow Charts	1	1	1	1

B Life Cycle Cost Plotting Program

In the early stages of this research effort it was obvious that it would be necessary to graphically display the data used. This program was the result of that need. It will read in the raw data in man-months per month, properly scale the axes, and plot the data versus time into the life cycle using squares at each data point. Figure B-1 shows a plot of the first set of data listed in Appendix A.

As the form of the function to model the software life cycle costs was being formulated, it was necessary to compare the plots of the actual data with that computed from the function. This capability was added in such a way that the program will plot any number of computed curves on the same axes as the actual data. This capability is demonstrated in Figure B-2.

The program listing is provided as it was implemented on a Control Data Corporation 6600 computer using available CALCOMP routines for on-line plotting. Figure B-3 shows a sample data deck for plotting actual data against two computed curves. Each sequential curve is given a different symbol to be plotted at each point from the table of symbols numbered 1 to 13 in the CALCOMP user Manual (Ref 24:47).

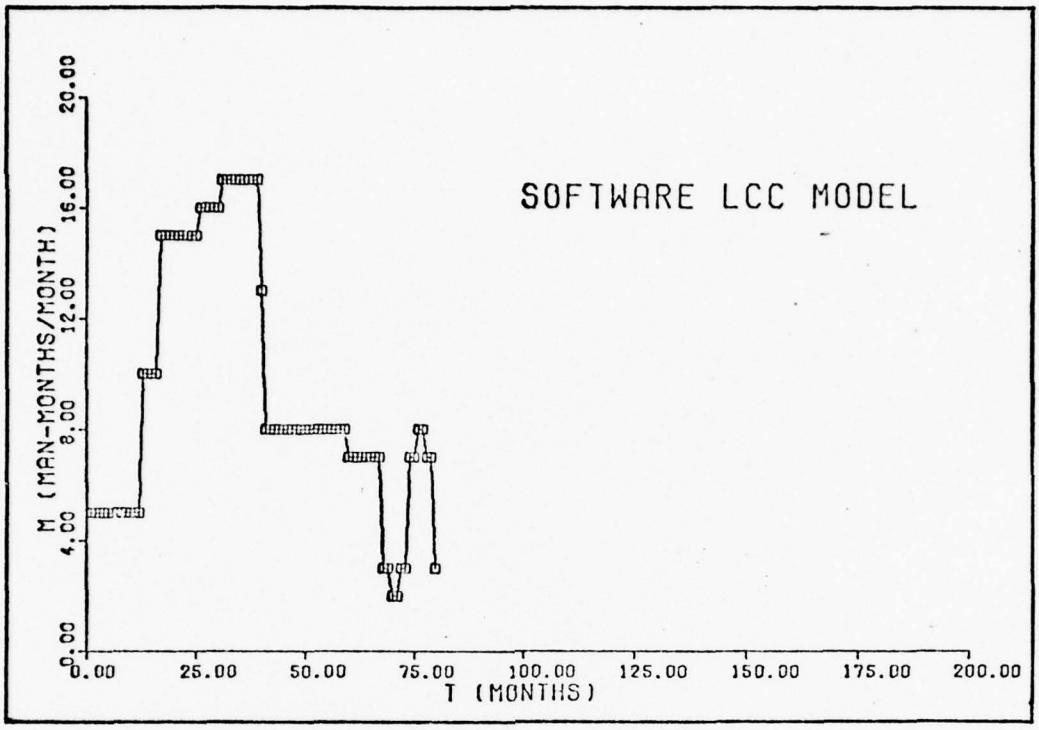


Figure B-1. Data Plot from LCCPLOT

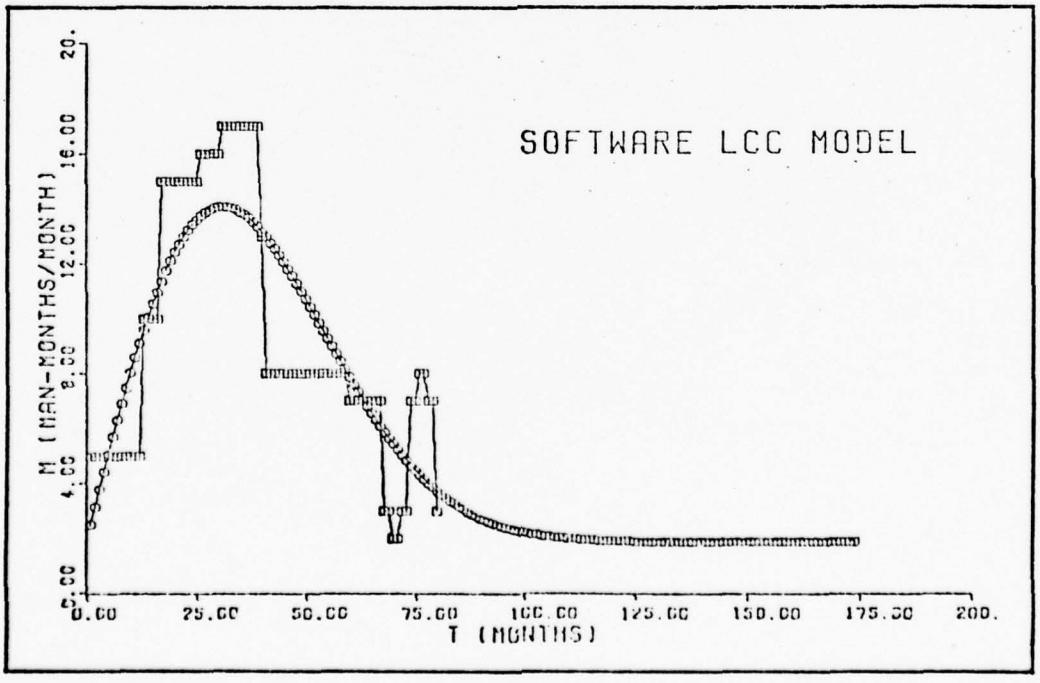


Figure B-2. Data and Computed Plots from LCCPLOT

```
80
5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,10.,10.,10.,10.,15.,15.,15.,15.,15.,15.,15.,
15.,15.,16.,16.,16.,16.,16.,17.,17.,17.,17.,17.,17.,17.,17.,17.,13.,8.,8.,8.,8.,
8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,7.,7.,7.,7.,7.,7.,7.,7.,   3.,3.,2.
,2.,3.,3.,7.,7.,8.,8.,7.,7.,3.
.8,.0035,2
.8,.0035,3
```

Figure B-3. Sample Data Deck for LCCPLOT

```
      PROGRAM LCCPLOT(INPUT,OUTPUT,TAPE4=INPUT,TAPE5=OUTPUT,PLOT)
C*************************************************************************
C***  THIS PROGRAM WILL PLOT INPUT RAW DATA AND/OR COMPUTED DATA FOR ***
C***  THE MODEL PRESENTED IN MY THESIS.  ANY NUMBER OF PLOTS MAY BE  ***
C***  DRAWN ON A SINGLE SET OF AXES.                                 ***
C*************************************************************************
      DIMENSION ACTUAL(202),T(202)
C
C  READ IN DATA
C
      READ *,NPTS
      IF(NPTS.EQ.0) GO TO 10
      READ *,(ACTUAL(I),I=1,NPTS)
      CALL SCALE(ACTUAL,5.,NPTS,1)
      AMIN=ACTUAL(NPTS+1)
      ASTEP=ACTUAL(NPTS+2)
      GO TO 20
10    READ *,AMIN,ASTEP
C
C  GENERATE AXES
C
20    CALL PLOT(1.,1.,-3)
      CALL AXIS(0.,0.,"M (MAN-MONTHS/MONTH)",20,5.,90.,AMIN,ASTEP)
      CALL AXIS(0.,0.,"T (MONTHS)",-10,8.,0.,0.,25.)
      CALL SYMBOL(4.,4.,.2,"SOFTWARE LCC MODEL",0.,18)
      IF(NPTS.EQ.0) GO TO 40
C
C  PLOAT RAW DATA POINTS
C
      DO 30 I=1,NPTS
30    T(I)=I
      T(NPTS+1)=0.
      T(NPTS+2)=25.
      CALL PLOT(0.,0.,3)
      CALL LINE(T,ACTUAL,NPTS,1,1,0)
```

```
C
C  GENERATE AND PLOT COMPUTED PLOTS
C
40     ISYM=1
       T(201)=0
       T(202)=25.
       ACTUAL(201)=AMIN
       ACTUAL(202)=ASTEP
       DO 50 I=1,200
50     T(I)=I
55     READ *,PK1,PK2,PK3
       IF(EOF(4).NE.0) GO TO 100
       DO 60 I=1,200
60     ACTUAL(I)=PK1*I*EXP(-PK2*I**2)+PK3
       CALL PLOT(0.,0.,3)
        CALL LINE(T,ACTUAL,200,1,1,ISYM)
       ISYM=ISYM+1
       GO TO 55
100    CALL PLOT(12.,0.,-3)
       STOP
       END
```

C Life Cycle Cost Model Program

This program was written to perform the tedious and complex computations used in the computational algorithm for building the model presented in this thesis. The program performs all the steps of the algorithm except selection of factors and iterating the third and fourth steps. The program reads in the data (manning and factors), fits function parameters to the data, solves the matrix equations for the model coefficients, calculates the sensitivities for each data set, and predicts the function parameters for systems from known factors.

Figure C-1 shows a sample data deck for use with this program. The following pages are a listing and sample output from the program using all four of the data sets listed in Appendix A and four factors. The predictions are for the same four data sets. The data set fitted parameters and predicted parameters are equal since the matrix equations are fully determined by the four data sets.

```
4
80,80
1.875
5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,10.,10.,10.,10.,15.,15.,15.,15.,15.,15.,15.,
15.,15.,16.,16.,16.,16.,16.,17.,17.,17.,17.,17.,17.,17.,17.,17.,13.,8.,8.,8.,8.,
8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,8.,7.,7.,7.,7.,7.,7.,7.,7.,   3.,3.,2.
,2.,3.,3.,7.,7.,8.,8.,7.,7.,3.
58,58
10.417
10.,13.,13.,15.,15.,15.,25.,25.,25.,34.,34.,34.,40.,40.,40.,45.,45.,45.,49.,49.,
49.,52.,53.,53.,56.,56.,56.,60.,60.,60.,66.,67.,67.,71.,72.,72.,73.,73.,73.,73.,
74.,74.,74.,74.,74.,74.,73.,73.,73.,73.,73.,55.,55.,55.,35.,35.,35.,35.
34,34
.554
5.,8.,8.,10.,12.,14.,16.,19.,20.,21.,22.,22.,22.,23.,22.,22.,22.,19.,17.,14.,13.
,12.,11.,10.,8.,7.,6.,4.,4.,3.,3.,3.,3.,3.
28,28
.274
2.,2.,3.,3.,4.,4.,4.,4.,4.,4.,4.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,5.,1.
,1.
4
% HUL      PROG QUAL DEV ON TGTMOD DESIGN
.38,39.0,1.,0.
.99,37.1,1.,0.
.53,62.8,0.,1.
1.0,82.4,0.,1.
.38,39.0,1.,0.
.99,37.1,1.,0.
.53,62.8,0.,1.
1.0,82.4,0.,1.
```

Figure C-1. Sample Data Deck for LCCMODL

```
      PROGRAM LCCMODL(INPUT,OUTPUT,TAPE4=INPUT,TAPE5=OUTPUT)
C
C
C*************************************************************************
C***  THIS PROGRAM IS USED TO IMPLEMENT THE MODEL OF LIFE CYCLE COST  ***
C***  OF SOFTWARE SYSTEMS PRESENTED IN MY THESIS.  THE INPUTS TO THE  ***
C***  PROGRAM ARE ACTUAL SOFTWARE COST DATA IN MAN-MONTHS EXPENDED    ***
C***  PER MONTH DURING THE LIFE CYCLE OF THE SYSTEM, THE FACTORS AND  ***
C***  THEIR VALUES FOR EACH OF THE DATA SETS, AND THE ASSUMED FACTORS***
C***  A PROPOSED SYSTEM TO BE PREDICTED.  THE OUTPUTS INCLUDE THE     ***
C***  FITTED PARAMETERS FOR EACH DATA SET, THE COEFFICIENTS OF THE    ***
C***  MODEL, THE SENSITIVITY ANALYSIS FOR EACH DATA SET, AND THE      ***
C***  PREDICTED PARAMETERS FOR THE UNKNOWN SYSTEM.                    ***
C*************************************************************************
C
C
      DIMENSION NPTS(5),NDPTS(5),ACTUAL(5,200),FK(5,3)
      DIMENSION IDFACTS(5),FACTORS(5,5),C(5,3)
      DIMENSION TK(5),TFACTS(5,5),PK(3),WRKAREA(50)
      DIMENSION SENS(5,5)
C
C    READ IN DATA SETS AND FACTORS
C
      READ *,NSETS
      DO 10 I=1,NSETS
      READ *,NPTS(I),NDPTS(I)
      IF(NPTS(I).EQ.NDPTS(I)) READ *,FK(I,3)
      N=NPTS(I)
10    READ *,(ACTUAL(I,J),J=1,N)
      READ *,NFACTS
      READ(4,11) (IDFACTS(I),I=1,NFACTS)
11    FORMAT(5A10)
      DO 20 I=1,NSETS
20    READ *,(FACTORS(I,J),J=1,NFACTS)
C
```

```
C    FIT PARAMETERS TO THE DATA
C
       DO 60 ISET=1,NSETS
       IMAX=1
       N=NPTS(ISET)
       DO 30 I=1,N
30     IF(ACTUAL(ISET,I).GT.ACTUAL(ISET,IMAX)) IMAX=I
       FK(ISET,2)=1./(2.*IMAX**2)
       IF(NPTS(ISET).EQ.NDPTS(ISET)) GO TO 45
       M=NDPTS(ISET)
       DO 40 I=M,N
40     FK(ISET,3)=FK(ISET,3)+ACTUAL(ISET,I)
       FK(ISET,3)=FK(ISET,3)/(N-M+1)
45     TOT=0
       DO 50 J=1,N
50     TOT=TOT+ACTUAL(ISET,J)
60     FK(ISET,1)=2.*FK(ISET,2)*(TOT-FK(ISET,3)*N)/(1.-EXP(-FK(ISET,2)*N
     *      **2))
C
C    SOLVE FOR THE MODEL COEFFICIENTS
C
       DO 90 KI=1,3
       DO 80 I=1,NSETS
       TK(I)=FK(I,KI)
       DO 80 J=1,NFACTS
80     TFACTS(I,J)=FACTORS(I,J)
       CALL LLSQAR(TFACTS,TK,NSETS,NFACTS,1,5,5,5,WRKAREA,IER)
       DO 90 I=1,NFACTS
90     C(I,KI)=TK(I)
C
C    PRINT OUTPUT REPORT
C
       WRITE(5,91) NSETS
91     FORMAT(1H1,///,10X,"SOFTWARE LIFE CYCLE COST MODEL"////,I3,
     *      " DATA SETS WERE USED.",//," THE FOLLOWING FACTORS WERE CONSI
```

```
     *DERED:")
      WRITE(5,92) (IDFACTS(I),I=1,NFACTS)
92    FORMAT(1H+,5(41X,A10,/))
C
C   PRINT OUT FITTED PARAMETERS
C
      WRITE(5,93)
93    FORMAT(///," THE FITTED PARAMETERS FOLLOW:",//," DATA SET",10X,
     *     "K1",10X,"K2",10X,"K3",/)
      DO 120 I=1,NSETS
120   WRITE(5,94) I,(FK(I,J),J=1,3)
94    FORMAT(1X,I5,2X,3(3X,G12.6),/)
C
C   PRINT OUT MODEL COEFFICIENTS
C
      WRITE(5,95)
95    FORMAT(///," THE DERIVED MODEL COEFFICIENTS FOLLOW:",//," FACTOR",
     *     13X,"AI",15X,"BI",15X,"CI",/)
      DO 130 I=1,NFACTS
130   WRITE(5,96) IDFACTS(I),(C(I,J),J=1,3)
96    FORMAT(1X,A10,3(5X,G12.6),/)
C
C   COMPUTE AND PRINT OUT THE SENSITIVITY ANALYSIS VALUES
C
      DO 150 ISET=1,NSETS
      WRITE(5,131) ISET,(IDFACTS(J),J=1,NFACTS)
131   FORMAT(////," DATA SET "I2" SENSITIVITY ANALYSIS BASED ON 200 MONT
     *H LIFE CYCLE."///" PARAMETER"5X"FACTORS",/,14X,5(A10,5X))
      EX=EXP(-FK(ISET,2)*40000)
      CLC=FK(ISET,1)*(1-EX)/(2*FK(ISET,2))+FK(ISET,3)*200
      DO 140 J=1,NFACTS
      SENS(J,1)=((1-EX)/(2*FK(ISET,2)))*C(J,1)/CLC
      SENS(J,2)=(-FK(ISET,1)*(1-EX)/(2*FK(ISET,2)**2)+FK(ISET,1)*40000
     *     *EX/(2*FK(ISET,2)))*C(J,2)/CLC
140   SENS(J,3)=200*C(J,3)/CLC
```

```
      DO 150 KI=1,3
150   WRITE(5,151) KI,(SENS(J,KI),J=1,NFACTS)
151   FORMAT(/,5X,"K",I1,6X,5(G12.6,3X))
C
C   PREDICT AND PRINT OUT UNKNOWN SYSTEM PARAMETERS
C
      WRITE(5,97)
97    FORMAT (    ,///," THE FOLLOWING PREDICTED PARAMETERS WERE COMPUTED
     *:",//," REQUEST",10X,"K1",15X,"K2",15X,"K3",/)
      NUM=1
100   READ *,(TK(I),I=1,NFACTS)
      IF(EOF(4).NE.0) GO TO 200
      DO 110 KI=1,3
      PK(KI)=0
      DO 110 I=1,NFACTS
110   PK(KI)=PK(KI)+C(I,KI)*TK(I)
      WRITE(5,111) NUM,(PK(I),I=1,3)
111   FORMAT(1X,I2,5X,3(5X,G12.6),/)
      NUM=NUM+1
      GO TO 100
200   STOP
      END
```

SOFTWARE LIFE CYCLE COST MODEL

4 DATA SETS WERE USED.

THE FOLLOWING FACTORS WERE CONSIDERED: % HOL
PROG QUAL
DEV ON TGT
MOD DESIGN

THE FITTED PARAMETERS FOLLOW:

DATA SET	K1	K2	K3
1	.649688	.520291E-03	1.87500
2	2.24167	.297442E-03	10.4170
3	2.20299	.255102E-02	.554000
4	.797770	.347222E-02	.274000

THE DERIVED MODEL COEFFICIENTS FOLLOW:

FACTOR	AI	BI	CI
% HOL	2.22064	-.203717E-03	12.9887
PROG QUAL	-.124945	.518852E-04	-.325748
DEV ON TGT	4.67870	-.142582E-02	9.64350
MOD DESIGN	8.87261	-.599397E-03	14.1270

DATA SET 1 SENSITIVITY ANALYSIS BASED ON 200 MONTH LIFE CYCLE.

PARAMETER	FACTORS % HOL	PROG QUAL	DEV ON TGT	MOD DESIGN
K1	2.13542	-.120150	4.49916	8.53212
K2	.244620	-.623027E-01	1.71209	.719745
K3	2.59942	-.651920E-01	1.92995	2.82724

DATA SET 2 SENSITIVITY ANALYSIS BASED ON 200 MONTH LIFE CYCLE.

PARAMETER	FACTORS % HOL	PROG QUAL	DEV ON TGT	MOD DESIGN
K1	.637919	-.358928E-01	1.34404	2.54882
K2	.441013	-.112322	3.08665	1.29759
K3	.443933	-.111336E-01	.329601	.482840

DATA SET 3 SENSITIVITY ANALYSIS BASED ON 200 MONTH LIFE CYCLE.

PARAMETER	FACTORS % HOL	PROG QUAL	DEV ON TGT	MOD DESIGN
K1	.802167	-.451342E-01	1.69010	3.20507
K2	.635498E-01	-.161856E-01	.444785	.186982
K3	4.78768	-.120072	3.55464	5.20728

DATA SET 4 SENSITIVITY ANALYSIS BASED ON 200 MONTH LIFE CYCLE.

PARAMETER	FACTORS % HOL	PROG QUAL	DEV ON TGT	MOD DESIGN
K1	1.88457	-.106036	3.97064	7.52984
K2	.397222E-01	-.101169E-01	.278015	.116874
K3	15.3097	-.383959	11.3668	16.6515

THE FOLLOWING PREDICTED PARAMETERS WERE COMPUTED:

REQUEST	K1	K2	K3
1	.649688	.520291E-03	1.87500
2	2.24167	.297442E-03	10.4170
3	2.20299	.255102E-02	.554000
4	.797770	.347222E-02	.274000

VITA

William H. Walker IV was born on 13 March 1950 in Minneapolis, Minnesota. He graduated from high school in Portland, Oregon, in 1968 and attended the United States Air Force Academy from which he received the degree of Bachelor of Science with two majors, computer science and mathematics, in June 1972. Upon graduation, he received a commission in the USAF and completed navigator training in April 1973. He then served as a C-141A airlift navigator, instructor navigator, and flight examiner navigator with the 14th Military Airlift Squadron, Norton AFB, California. He was selected to enter the School of Engineering, Air Force Institute of Technology, in June 1977.

Permanent address: 2120 Highway 101 North
Rockaway, Oregon 97136

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GCS/EE/78-21	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE *(and Subtitle)* AN APPROACH TO SOFTWARE LIFE CYCLE COST MODELING		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) William H. Walker IV Capt USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology(AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (RADC/ISIS) Griffiss AFB, New York 13441		12. REPORT DATE December 1978 13. NUMBER OF PAGES 75
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT *(of the abstract entered in Block 20, if different from Report)*

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

JOSEPH P. HIPPS, Maj, USAF
Director of Information 1-23-79

19. KEY WORDS *(Continue on reverse side if necessary and identify by block number)*

Life Cycle Cost
Modeling

20. ABSTRACT *(Continue on reverse side if necessary and identify by block number)*

This report describes the development of a software life cycle costing model. The model reduces life cycle cost to a function of three parameters which are, in turn, functions of a number of factors that describe the software system. A step-by-step algorithm is presented for building the model from raw data. The model is exercised as an example with a small amount of data. The most salient factors are selected by use of sensitivity

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

analysis. Brief descriptions of management applications and recommendations are presented. Appendices describe sample data and two computer programs used to develop the model.